# CS664 Lecture #27: Some Topics We Didn't Cover

**Some material taken from:**

- **Yuri Boykov & Olga Veksler, University of Western Ontario**
- **Uri Feige, Weizmann Institute & Microsoft Research**

# Reminders

- PS3 is due Friday 12/2
- Papers will be graded by tomorrow
  - I will email you about how to pick them up
- Please fill out course evaluations
  - In particular, I'd like to hear about:
    - Choice of papers we covered/didn't cover
    - Effectiveness of the quizzes
- Final project is due Thursday 12/15
  - The next slide is from Lecture 1, 8/25/05

# Final Project

- *Most of the course grade comes from this*
  - *It is actually possible to get an F in CS664*
- *You need to pick a project by midway through the term, and work hard on it*
  - *Write-up due during final exam period*
  - *Proposal due by November 1*
- *Must be original research*
  - *i.e., non-trivial idea that has never been done*
- *Not required to solve a vision problem*
  - *Document idea's strengths and weaknesses*

# Today's Topics

- Particle filters
- Learning low-level vision
- Segmentation via graphs

# 1. Particle filtering

- Another alternative is to represent the posterior distribution by a set of particles
  - Points with probabilities (point masses)
- Two phases: prediction, update
  - Prediction: each particle is modified according to a “motion model”
    - Including random noise
  - Update: weights re-evaluated using the new observed data

# Importance sampling

- We have samples from the prior distribution p(x), and the likelihood p(z|x)
  - We need to sample from posterior p(x|z)
  - More generally, given samples from g(x), how do we create samples from h(x)?
- We can rescale the weight of a sample from g(x) by the ratio h(x)/g(x)
  - Then we normalize
  - In our example, we scale up by the likelihood
    - I.e., multiply a particle $x_i$'s weight by the likelihood $p(z|x_i)$

# Particle filtering issues

- With enough particles you can get a very good estimate of the posterior
  - But it may not be tractable
  - Especially in high-dimensional spaces
- Example difficulty: particles can easily disappear (resampling problem)
- Generally, a successful approach, especially for hand tracking

# 2. Learning low-level vision

- Super-resolution problem (Astronomy)
  - Create a high-resolution image from a low-resolution one
  - Obviously this is ill-posed, but it's sometimes possible with enough prior information
  - A nice problem from a Bayesian perspective
- In our low-resolution image, we have an observed intensity $o_i$ at each pixel
  - This is created from a high-resolution true image $h_i$ by averaging plus noise

# Training data

- We can take a bunch of high-resolution images and create low-resolution versions
- Based on this, we will create our labels
  - A set of hypotheses $\{h_i\}$ for each intensity $o_i$
- Need a prior that relates neighboring hypotheses $h_i, h_j$
- Can make two hypotheses overlap slightly
  - Determine compatability by the difference in intensities in the overlap
  - Can estimate parameters from training data

# Examples

(a) input

(b) desired output

Cubic splines

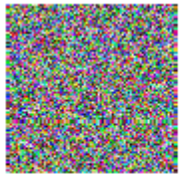
Learning

# Output depends on training

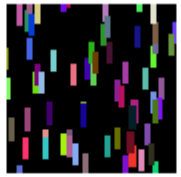
(a) Actual

(b) Input

(c) Cubic spline



(d) noise



(e) rectangles

(f) generic

(g) Train: noise

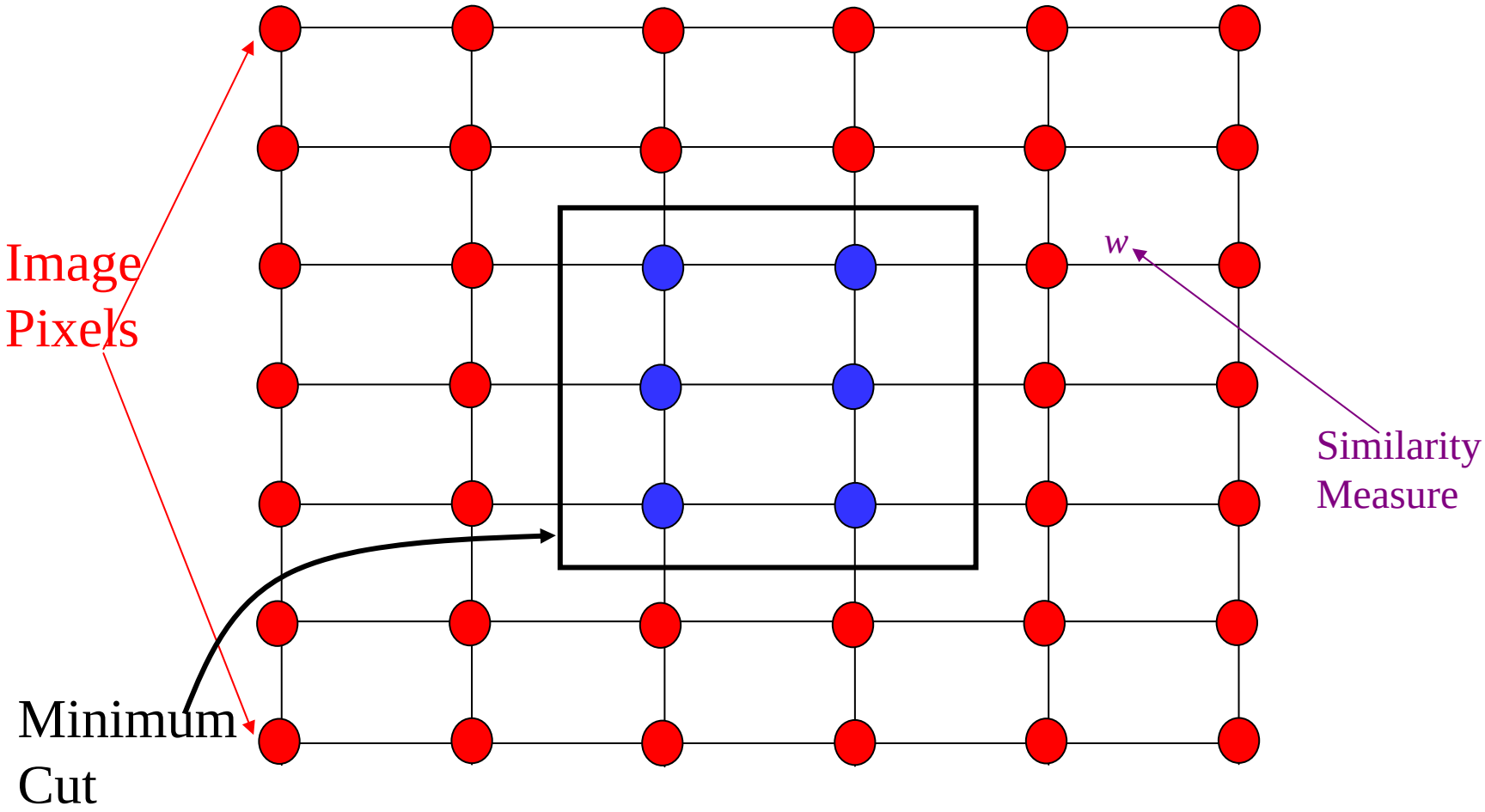
(h) Train: rects

(i) Train: generic

# 3. Graph-based Segmentation

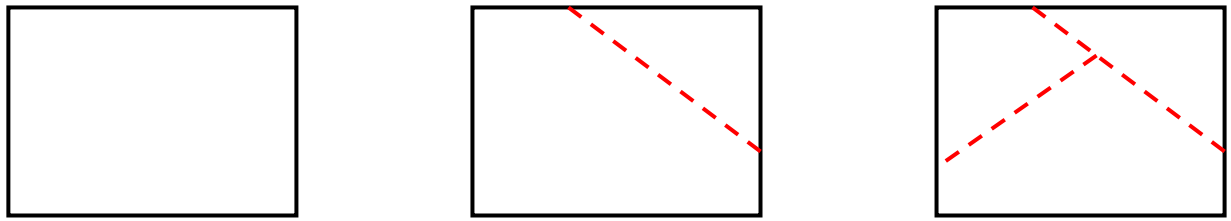
- Why can't we use min cut to directly segment an image?
  - Weights between edges are "affinities"
    - I.e., decreasing function of $|I_p - I_q|$
  - Where do the source and sink go?
- Two answers
  - Use variant on min cut where there is no source and sink
    - E.g., Karger's algorithm
  - Nested cuts

# Segmentation via min cut

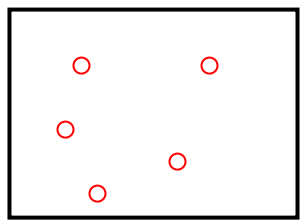


# Wu and Leahy

- Recursively split the image in 2
  - Each stage is fast



- Problems:
  - Image may have no natural binary split!
  - Bias towards small segmentations



# Normalized cuts

- We get in trouble by minimizing

  $\text{cut}(A,B) = \sum_{i \in A, j \in B} w_{ij}$

- We can normalize this by using

  $\text{assoc}(A,S) = \sum_{j \in A} \sum_{i \in S} w_{ij}$

  – assoc(A,A) measures the association within A

- A "normalized" measure of how similar A and B are within themselves is:

$$\text{Nassoc}(A,B) = \frac{\text{assoc}(A,A)}{\text{assoc}(A,V)} + \frac{\text{assoc}(B,B)}{\text{assoc}(B,V)}$$

# Normalized cut criterion

- Consider the normalized cut measure

$$\text{Ncut}(A,B) = \frac{\text{cut}(A,B)}{\text{assoc}(A,V)} + \frac{\text{cut}(A,B)}{\text{assoc}(B,V)}$$

  - This normalizes the value of cut(A,B)
  - Minimizing Ncut usually avoids small partitions
    - Because assoc(A,V) is small if A is small
- Can show Ncut(A,B) = 2 – Nassoc(A,B)
  - So we can simultaneously find a cut that avoids small partitions while maximizing similarity within A and B

# Some eigenvector facts

- An eigenvector x with eigenvalue $\lambda$ obeys:

$$Wx = \lambda x$$

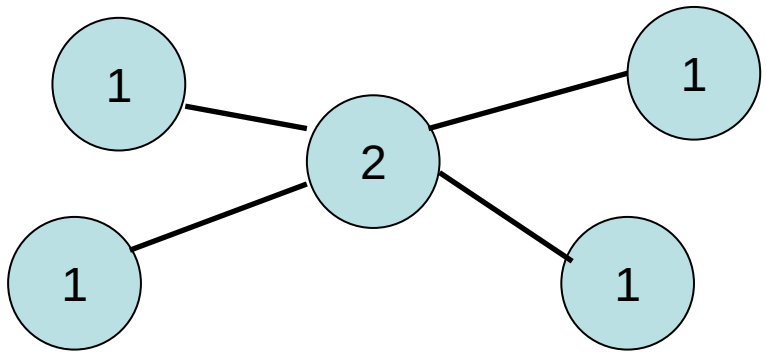
  - Note that a multiple of x is also an eigenvector
  - There can be multiple different eigenvectors with the same eigenvalue $\lambda$
    - The multiplicity of $\lambda$
- If W is symmetric:
  - Eigenvalues are real
  - Eigenvectors are orthogonal

# Spectral graph partitioning

- Consider a graph where all $w_{ij} = 1$
  - Can use multiple edges for arbitrary weights
  - Also, assume all nodes have degree d
  - Assign the weight $x_i$ to the node i
- What is an eigenvector of W?

$$\sum_{j:(i,j)\in E} x_j = \lambda x_i$$

- Example for $\lambda = 3$



# Eigenvector properties

- The largest eigenvector for this graph has eigenvalue $\lambda = d$
  - One eigenvector is **1**
    - All other eigenvectors are orthogonal to **1**
    - So their entries sum to 0
- The set of eigenvalues of W is called the graph's spectrum
  - The 2nd largest eigenvector can be used for *spectral graph partitioning*
- Example: bisectable graphs (|A|=|B|)

# Spectral graph bisection

- An eigenvector with $\lambda = d$ assigns one component +1 and the other -1
- If each node has $\varepsilon d$ neighbors, this is still an eigenvector with $\lambda = d(1-2\varepsilon)$
- Instead of the adjacency matrix, often use the graph's Laplacian L=D-W
  - D(i,i) = $\sum_j w_{ij}$ is the degree matrix (diagonal)
  - Seek the 2nd smallest eigenvalue of L instead of the 2nd largest of W
  - This eigenvector is called the Fiedler vector

# Computing Normalized Cuts

- Minimizing Ncut is equivalent to the integer programming problem: minimize

$$\frac{y^T (D-W)y}{y^T D\, y}$$

- W is the affinity matrix
- Minimize under: $y_i \in \{0,1\}$ and $y^T D\mathbf{1} = 0$
- Close to spectral graph partitioning
  - Beware of integrality constraints!

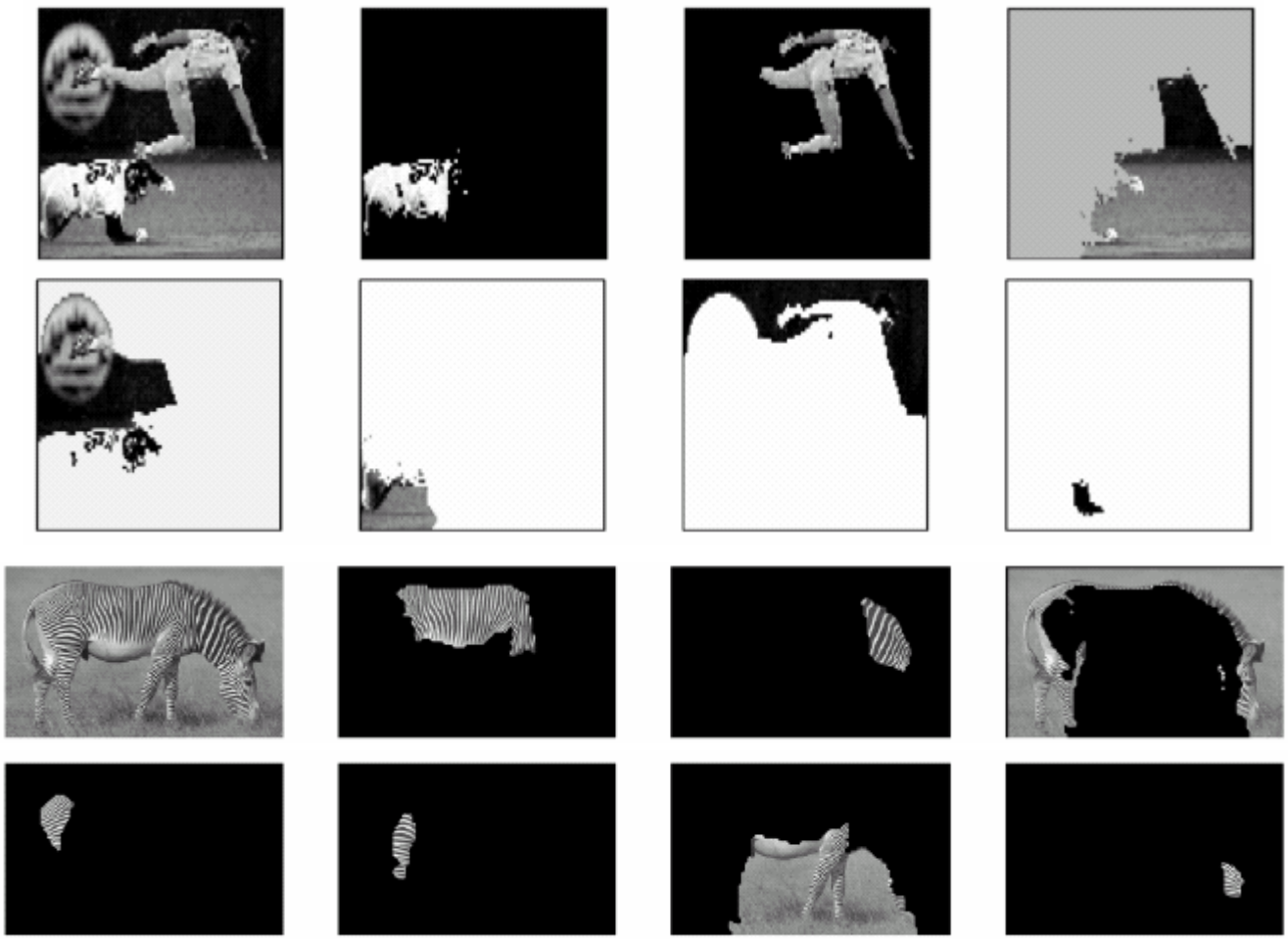
# Eigenvector approximation

- Integer programming problem NP hard
  - Instead solve continuous (real-valued) version
  - This corresponds to finding second smallest eigenvector of

$$(D-W)y_i = \lambda_i Dy_i$$

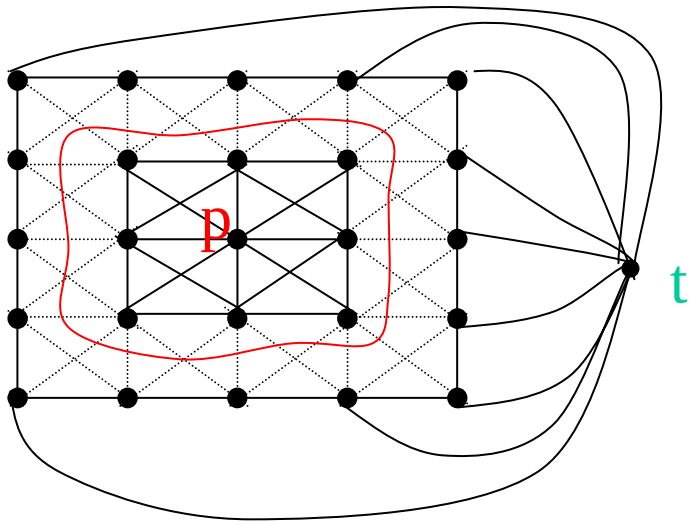
- Widely used method
  - Works well in practice
    - Large eigenvector problem, but sparse matrices
    - Often resolution reduce images, e.g, 100x100
  - But no longer clearly related to cut problem

# Normalized Cut Examples

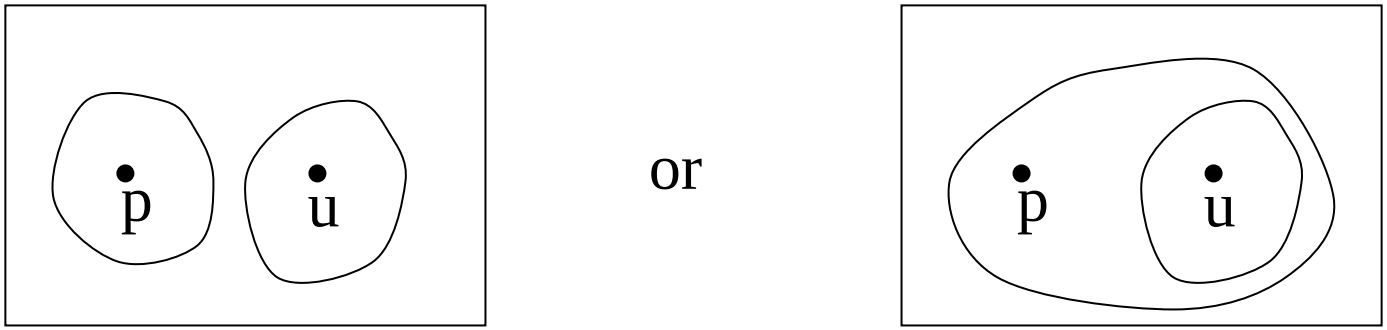


# Nested cuts

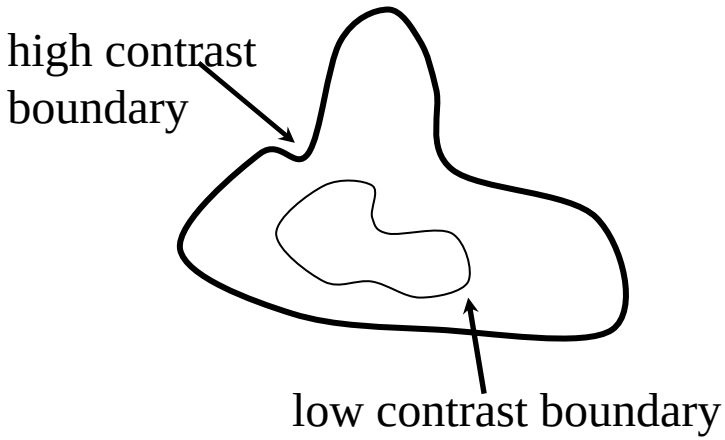
- Each pixel should be in a “good” region
  - Compute an p-t min cut for each pixel p
    - t will lie outside the image boundary

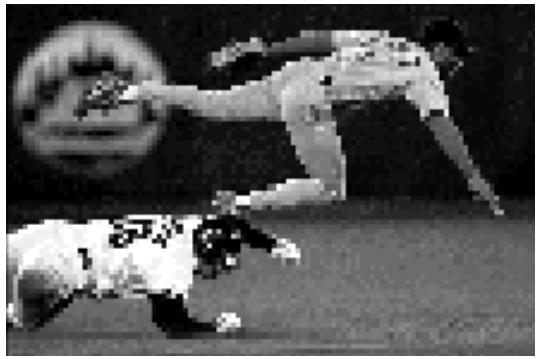


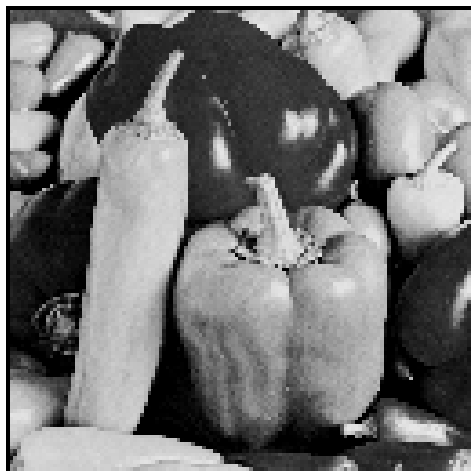
# Properties

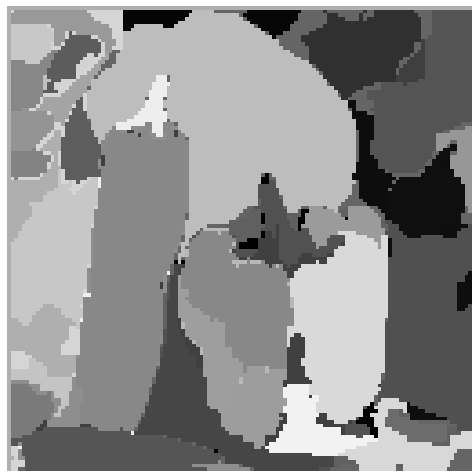



Recurse for each region:





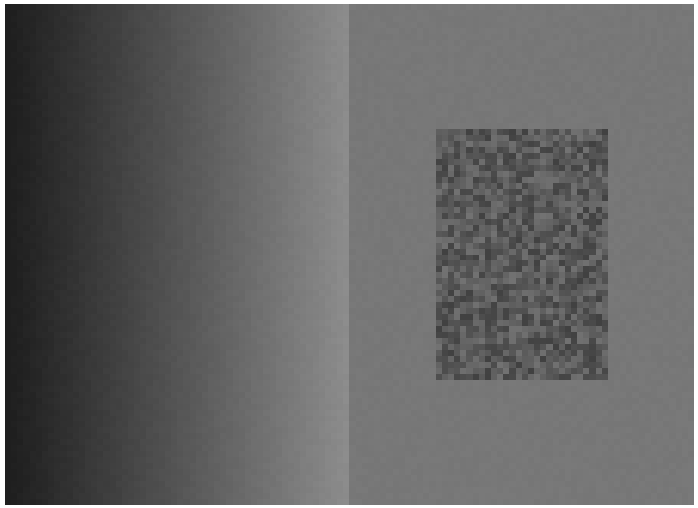




# MST-based segmentation

- Minimum spanning tree is the cheapest way to connect all pixels into a single component (or “region”)
  - Merge two components when the cheapest edge between them is cheap compared to a measure of the internal variation
- Provably good segmentation under a fairly natural definition
  - Neither too coarse nor too fine

# MST segmentation examples



# Conclusions

- What have we learned?
  - Lots of interesting techniques can be applied
  - Some of them work, at least some of the time
  - Vision is still hard
    - But, the problems are very important
    - And the field does in fact make progress!